ISICSoo: a class for the calculation of ionization cross sections from ECPSSR and PWBA theory

Matej Batič[a,b,*], Maria Grazia Pia[a], Sam J. Cipolla[c]

[a]*Instituto Nazionale di Fisica Nucleare, Sezione di Genova, Via Dodecaneso 33, 16146 Genova, Italy*
[b]*Jožef Stefan Institute, Jamova 39, 1000 Ljubljana, Slovenia*
[c]*Physics Department, Creighton University, Omaha, NE 68178, USA*

Abstract

ISICS, originally a C language program for calculating K-, L- and M-shell ionization and X-ray production cross sections from ECPSSR and PWBA theory, has been reengineered into a C++ language class, named ISICSoo. The new software design enables the use of ISICS functionality in other software systems. The code, originally developed for Microsoft Windows operating systems, has been ported to Linux and Mac OS platforms to facilitate its use in a wider scientific environment. The reengineered software also includes some fixes to the original implementation, which ensure more robust computational results and a review of some physics parameters used in the computation. The paper describes the software design and the modifications to the implementation with respect to the previous version; it also documents the test process and provides some indications about the software performance.

Keywords: Atomic K-, L- and M-shell ionization cross section, PWBA cross sections, ECPSSR, ISICS

NEW VERSION PROGRAM SUMMARY

Program Title: ISICSoo
Journal Reference:
Catalogue identifier:
Licensing provisions: Standard CPC licence, `http://cpc.cs.qub.ac.uk/licence/licence.html`
Programming language: C++
Computer: 80486 or higher-level PC or Mac
Operating system: any OS with gcc compiler version 4.1 (or newer); tested on Scientific Linux 5 (gcc 4.1.2), Mac OS X 10.6.5 (gcc 4.2.1) and Windows XP (MS Visual C++ 2010 Express)
Classification: 16.7
Catalogue identifier of previous version: ADDS_v4_0
Journal reference of previous version: Comput. Phys. Comm. 180 (2009) 1716
Does the new version supersede the previous version?: no
Nature of problem: Ionization and X-ray production cross section calculations for ion-atom collisions.
Solution method: Numerical integration of form factor using a logarithmic transform and Gaussian quadrature, plus exact integration limits.
Reasons for the new version: Capability of using ISICS physics functionality in other software systems; porting the software to other platforms than Microsoft Windows; improved computational robustness and performance.
Summary of revisions: Reengineering into a C++ class; several internal modifications to improve correctness and robustness; updated binding energies tabulations; performance improvements.
Running time: The running time depends on the selected atomic shell and the number of polynomials used in the Gaussian quadrature integration.

---

*Corresponding author.
E-mail address: matej.batic@ge.infn.it

1. Introduction

ISICS (Inner-Shell Ionization Cross Sections) [1, 2, 3, 4] is a C language program that computes ionization and X-ray production cross-sections for K-, L-, and M-shells in the ECPSSR [5] and PWBA [6] theoretical frameworks, using exact integration limits for calculating the form factors. Both theories have found use in numerous applications, such as those involving elemental composition analysis or ion beam transport, and research involving projectile charge change and energy loss, recoil ion production, or target vacancy rearrangement and X-ray production.

A few other open source programs calculating ECPSSR cross sections can be found in the literature. Šmit's code [7] calculates cross sections for K- and L-shells and is implemented in Pascal. The ERCS08 program [8], implemented in FORTRAN, calculates cross section for K-, L- and M-shell ionization according to various options of the ECPSSR theory. Both of them, as well as ISICS, are stand-alone applications; as such, they are not meant to be used within another application. All these codes run on Windows platforms; ERCS08 can run with minor modifications on other platforms equipped with a FORTRAN compiler, but its graphical user interface (GUI) is specific to the Windows environment.

ISICSoo addresses these shortcomings: its main motivation is the capability of using ISICS physics functionality in other software systems, rather than only as a stand-alone program, and in a variety of platforms, not limited to the original Microsoft Windows environment. For this purpose, ISICS has been reengineered into a C++ class, which preserves the same physics functionality as the original stand-alone ISICS program, while providing greater flexibility of use and improved robustness of implementation.

Although the physics capabilities of ISICSoo are equivalent to those of ISICS, the reengineering process determined major changes in the software architecture and design, which have significant implications on the use of its physics functionality in scientific applications.

The reengineered version facilitates the exploitation of ISICS cross section calculations for large scale productions requiring a variety of settings, such as the tabulation of data libraries; it also enables the exploitation of ISICS physics functionality within general-purpose Monte Carlo systems for particle transport and PIXE (Particle Induced X-ray Emission) analysis systems. More in general, the new form of ISICS is suitable to applications where a versatile, platform-independent tool for the calculation of inner shell ionization cross sections by proton or ion impact is needed. As such, it complements the original ISICS Windows version, which is oriented for small scale individual use.

The following sections outline the software design and the improvements to the implementation available in this new development of ISICS; they also document the use of the ISICSoo class and the test process it has been subject to prior to its release in the CPC Library.

The reengineering of ISICS is based on its latest published version, ISICS 2008 [3], complemented by an unpublished ISICS 2010 version [9], which includes a few small fixes to the 2008 version. The changes in cross section calculations originally implemented in ISICSoo, described in section 4.2, were successively implemented also in the later ISICS 2011 version [4].

2. Software design

The following sections provide an overview of relevant issues addressed by the reengineering process and the resulting new features of the software design.

2.1. The refactoring process

Reengineering [10] is a process of examination and alteration of an existing software system to reconstitute it in a new form. It exploits established techniques and patterns, that embody knowledge and best practices adopted by the software community.

The reengineering of ISICS addressed various issues, including:

- unbundling the monolithic ISICS system, so that some individual parts of it – namely, its cross section calculations – could be used independently;
- porting the system to other platforms than Microsoft Windows: this process required reworking the software architecture to clearly separate the platform-dependent code;

- the absence of layering in the original procedural code hindered portability and adaptability: the reengineering analysis identified two layers - database and user interface - along with the proper physics domain;
- the shift from the procedural programming paradigm of ISICS to the object oriented paradigm.

To a large extent, the final stage of the reengineering process consisted of refactoring [11], i.e. a process that changes the structure of a software system without changing its observable functionality. Refactoring methods specific to converting procedural code to objects [11] were applied.

2.2. Design features

The adoption of the object oriented technology for the reengineering of ISICS ensures a versatile, yet compact design. A single class, named ISICSoo, is responsible for the physics functionality originally provided by ISICS and for the interface with the client.

In line with best practices of object oriented design [12, 13, 14, 15], ISICSoo responsibility is focused on cross section calculation; other responsibilities in response to user requirements (such as, for instance, error handling) are expected to be provided by other entities (classes or packages) of the user application or framework.

Consistently with the object oriented paradigm, an ISICSoo object encompasses functions associated with its responsibilities and the data needed for fulfilling them.

The class constructor, invoked when an instance of the ISICSoo class is created, is responsible for configuring the state of the instantiated object with a set of default options, including the physics data to be used in the calculations.

The public class interface is minimal, being limited to operations pertaining to interactions with the client, while the physics functionality of the object is relegated to private member functions. Public member functions set-up the calculation environment, trigger the calculation of the requested cross sections and retrieve the results of the computation. All data members are private.

The `SetVerbosity` member function controls the textual output from the ISICSoo object; it addresses different use cases, from batch running in production mode to minute monitoring of the calculation. Its integer argument defines the desired level of detail: `SetVerbosity(0)` suppresses all output, `SetVerbosity(1)` will print concise information, while `SetVerbosity(2)` will print exhaustive details.

A Doxygen [16] documentation, illustrating the interface, data members of the ISICSoo class and some examples, is included in the CPC package.

3. Client interface and computation configuration

The class interface defines how a client can interact with an ISICSoo object; it consists of functions for setting the parameters and the input data to be used in the physics calculations.

In addition, the occurrence of anomalies in the execution of operations of the ISICSoo class is signaled to the user by logging informative error messages to the standard error stream `std::cerr`, which the user can redirect to a file, if desired.

3.1. Configuration settings

A number of parameters must be specified to set-up the computation environment prior to performing cross section calculations: the characteristics of the projectile (particle type and energy), the target atom, the shell (or shells) for which the cross sections are to be calculated, and the modeling approach to be activated (PWBA or ECPSSR, the latter in turn articulated over a few different options).

The computation settings can be defined either through an external configuration file, similarly to the original ISICS program, or through public member function calls. The functionality for parsing configuration files is delegated to private member functions. Configurations settings defined through the ISICSoo public interface can be saved to files for subsequent reuse. The class interface provides convenient functionality to generate a large variety of configurations, as may be needed for large scale productions (e.g. to tabulate a cross section data library). The format of the configuration files is compatible with the original ISICS 2008 version to facilitate the reuse of existing settings in different environments.

ISICSoo provides the same ECPSSR options as the ISICS program: the original formulation of the theory [5] (identified as "plain ECPSSR" in the following), the United-Atom approximation [17], the Hartree-Slater description of K-shell electrons [18] and the treatment of K-shell ionization by relativistic protons [19].

The calculation of cross sections for a given shell can be activated by invoking the associated member function with the appropriate argument (e.g.`CalculateKShell(true)`).

The projectile and the target can be specified either by passing the atomic number Z or the name of the element as an argument to the member functions responsible for their definition.

In the reengineered version it is now possible to calculate the cross sections at a single energy, at equidistant values within a given energy range or by providing a list of predefined energies.

Default values are attributed in the class constructor to all the options selectable through the ISICSoo public interface. This feature, which complies with good C++ programming practice, avoids the risk that some variables – namely data members – may remain uninitialized. The public class interface lets the user override the default values.

Default values provide a standard configuration of ISICSoo objects; however, it is worthwhile to stress that optimal options for given projectile and target parameters do not exist in absolute terms, but depend on the use case.

Moreover, differently from the limited spectrum of use of the stand-alone ISICS program, several instances of the ISICSoo class may coexist in the same application, each one characterized by different option settings: examples of such use cases are a data library production, or the investigation of different cross section models.

Such flexibility is a relevant feature of the shift to the object oriented paradigm and of the design of ISICSoo as a class, with respect to the nature of procedural program of the original ISICS. The results of ISICSoo validation process provide guidance for the selection of options suitable to different experimental scenarios.

3.2. Physics parameters

The ISICSoo class uses various physics data in the calculation of cross sections, such as masses of elements, atomic electron binding energies, fluorescence and Coster-Kronig transition parameters.

The same data sets are supplied along with the source code as in the previous version of ISICS [3]; the ISICSoo class can load alternative data supplied by the user, provided the data files are formatted in the same way as the original ones.

The data sets are expected to be found by default in the same directory where ISICSoo resides; alternatively, the user can place them in a different directory, whose path is specified through the `ISICS_DATA` environmental variable. An example of how to load user-supplied data is given in the code listing 1.

4. Features of ISICSoo implementation

4.1. Generalities

The ISICSoo class is implemented in C++; the choice of this language allows the new version to profit from the benefits of the object oriented paradigm, while the compliance of C++ syntax with C facilitated refactoring the original C code. Indeed, large part of the original ISICS C language code has been kept as is in the new class implementation, changing the code only where necessary to reflect the new software design.

The implementation of ISICSoo is compliant with C++ standard [20] and uses the Standard Template Library, making the code highly portable to different platforms.

In the process of migrating to C++, the precision of the calculation has been changed from single precision (`float`) to double precision (`double`), since double precision arithmetic is nowadays considered a standard in mathematical calculations [21].

The emphasis of this paper is on the calculation of ionization cross sections; nevertheless ISICSoo also implements the calculation of X-ray production cross sections for consistency with the legacy ISICS code. According to the iterative-incremental software development process adopted by ISICSoo, based on the Unified Process [22] framework, the responsibility for X-ray production is intended to be delegated to a collaborating class (or package) in a future version.

4.2. Changes in cross section calculations

ISICSoo encompasses a few changes concerning the calculation of ECPSSR cross sections with respect to the previous ISICS 2008 version, which contribute to the robustness of the results and improve the computational performance.

Two of such changes had first been incorporated by the original author of the ISICS program into ISICS 2010 version, which was downloaded from the author's web page [9]. They prevent the generation of unphysical results if the electron binding energy for a given shell happens to equal zero in the data set used by ISICS, or the quantity

$$1 - \frac{4}{M\zeta_S\theta_S}\left(\frac{\zeta_S}{\xi_S}\right)^2 \tag{1}$$

which appears as the argument of a square root in the calculation of z_s (equation A.8 in [1]), is negative; in both cases the cross section values are set to be zero. The upper energy limit, below which the quantity in formula (1) becomes negative, is illustrated in Figure 1 as a function of the target element for different ionized shells and protons and α particles as projectiles; one can observe that the occurrence of unphysical negative values concerns relatively low energies with respect to typical experimental applications of PIXE techniques, which represent the main domain of interest for ISICS.
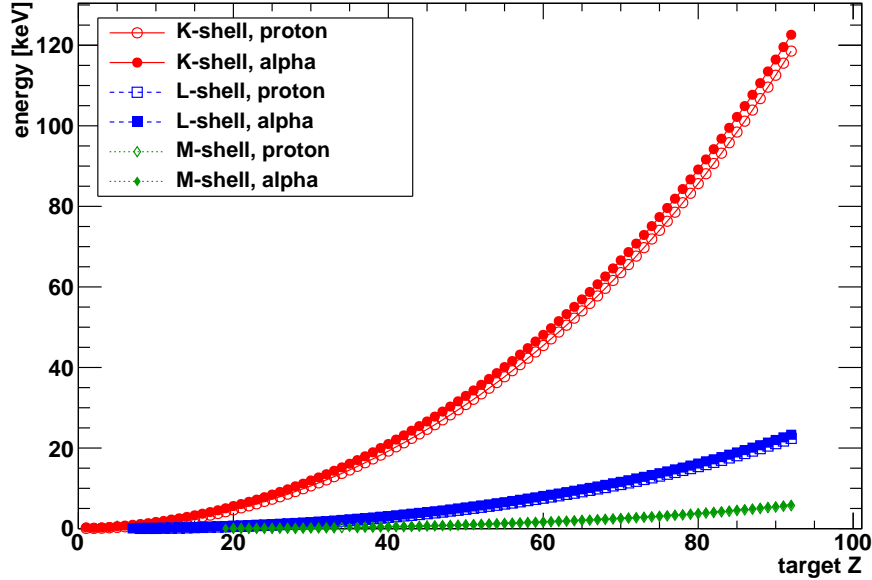

Figure 1: Energy limit, below which negative values occur in the expression of (1), as a function of the target element for different ionized shells and projectiles as indicated in the legend.

The following other changes are specific to the ISICSoo implementation and are not implemented in either 2008 or 2010 version of ISICS.

The test process adopted in the software development identified the occurrence of unphysical negative limits passed to the integration algorithm in two cases, which result in ECPSSR cross sections appearing as `nan` (not-a-number) in previous versions of ISICS. One of such cases concerns a few light elements at low projectile energies, when the ζ_S term of equation A.2 in [1] is negative. The number of affected elements and energy range depends on the projectile; for proton as projectile ζ_S is negative below ~ 20 keV for oxygen and fluorine as target materials.

The other case occurs when the binding energy for a given shell of the combined target and projectile system (i.e. corresponding to an atom whose atomic number is the sum of the atomic numbers of the two) happens to be zero in the binding energy tabulations used by ISICS. This anomaly has been observed, for instance, in M-shell ionization by proton impact on bromine ($Z = 35$), since the binding energy tabulations

used by ISICS report a value of zero for the M_1 binding energy of krypton ($Z = 36$). The ISICSoo class returns zero as the value of the ECPSSR cross section in both such cases.

In the special case of proton incident on bromine, the atomic binding energy tabulation distributed with ISICSoo in the CPC Library has been updated to include a positive value of the M_1 binding energy of krypton, derived from [34]. Special care should be taken to ensure that user supplied binding energy tabulations provide positive binding energy values not only for the target selected for calculation, but also for the element corresponding to the composite target and projectile system.

Whenever, due to intrinsic deficiencies of the theory, numerical artifacts or whatever other reasons, the cross section calculation algorithm would result in unphysical values, the ISICSoo member functions responsible for ECPSSR and PWBA cross section calculation return zero value. Therefore, the objects interacting with ISICSoo instances can be aware of the occurrence of unphysical conditions, while dealing with such occurrences properly left to their responsibility. The occurrence of unphysical cross section values is logged to the standard error stream (`std::cerr`).

It is worthwhile to stress that, differently from ISICSoo, which is a stand-alone program, ISICSoo is a class and, as such, must have a single, cohesive responsibility; other responsibilities in response to user requirements (such as, for instance, error handling) are expected to be provided by other entities – classes or packages – of the user application or framework.

Other modifications with respect to the previous version of ISICS concern the consistency of calculation of the available options of the ECPSSR theory. These verifications prevent the inappropriate application to the L-shell of the scaling function associated with the correction for the relativistic Dirac-Hartree-Slater nature of K-shell, which occurred in the previous version of ISICS.

Improved computational performance is achieved in ISICSoo by separating the calculation of PWBA and ECPSSR cross sections in dedicated functions. This modification prevents the unnecessary duplication of the PWBA calculation, which occurs in the previous version of ISICS when ECPSSR cross sections are calculated, and the redundant calculation of ECPSSR cross sections, when only PWBA ones are desired.

As a further contribution to optimize the performance, special care has been taken to ensure that the calculation of weights and abscissas for Gaussian quadrature integration is performed just once, when the number of polynomials is chosen through the client interface.

A new version of ISICS for Windows (ISICS 2011) [4] has been released in the CPC Program Library after the submission of the present paper to Computer Physics Communications. Its functionality is equivalent to ISICSoo, as it incorporates the aforementioned changes implemented in the ISICSoo class on top of the 2010 unpublished version.

5. Tests of the new version

The reengineered version of the code has been subject to rigorous testing, which includes the verification of the result of the reengineering process and validation with respect to experimental measurements.

A simple application was developed for the purpose of testing, which instantiates an object of ISICSoo type and uses it to calculate cross sections corresponding to various modeling options in predefined combinations of projectile and target parameters.

5.1. Verification

The verification process involved the comparison of results calculated by the ISICSoo class with the outcome of the original ISICS. Consistently with the IEEE Standard for Software Verification and Validation [23], the regression testing comprised in the verification process was performed with respect to previous versions of the code: the published ISICS 2008 version and the 2010 one, which can be downloaded from the ISICS author's personal web site.

Apart from being the latest published version, ISICS 2008 was used to produce a cross section data set distributed with Geant4 [24, 25] 9.4 version; therefore consistency of results is an important verification for future data library productions based on ISICSoo. ISICS 2010, although not released in the CPC Program Library, was the original version in the reengineering process that led to ISICSoo and verification of consistency with it was regularly monitored in the course of the refactoring process.

The test concerned the plain ECPSSR K-shell ionization cross sections by proton on elements with atomic number between 2 and 92, calculated for a set of energies between 100 keV and 100 MeV. The calculations for elements with missing or wrong binding energies data in the ISICS 2008 version (e.g. europium) were excluded; similarly, the calculation cases affected by the modifications specific to ISICSoo described in section 4.2 were not considered in the regression tests with respect to ISICS 2010.

The relative differences between the cross section values calculated by ISICSoo and ISICS (either 2008 or 2010) are below the machine precision in 98.3 ± 0.1% of test cases. The remaining 1.7 ± 0.1% of test cases exhibit relative differences up to approximately 10^{-3}, attributed to the different precision of calculation mentioned in section 4.1 and difference in number of significant digits in the output between ISICS and ISICSoo. All such differences are by far smaller than the experimental uncertainties by which inner shell ionization cross sections are measured.

As a result of the verification process, one can conclude that ISICSoo cross section calculations are consistent with those of previous ISICS versions; therefore, apart from the improvements described in 4.2, the reengineering process has retained the same correctness as the original ISICS physics output.

It is worthwhile to remind the reader that the later ISICS 2011 version is identical to ISICS 2010, apart from incorporating the changes described in section 4.2 originally implemented in ISICSoo. Therefore, although reference [4] does not explicitly document ISICS 2011 regression testing, on the basis of the above verification process one can infer the equivalence of physics results between ISICSoo and the later released ISICS 2011 version.

5.2. Physics validation

Software validation is an essential part of the software development process and a key component in the assessment of the quality of a software product. The validation of ISICSoo followed the guidelines of the IEEE Standard for Software Verification and Validation [23].

The validation process involved the comparison of cross sections by proton impact calculated by ISICSoo with experimental measurements [26, 27, 28]. This evaluation of compatibility of the cross sections calculated by ISICSoo with experimental data provides a quantitative appraisal of the accuracy achievable in applications of the code.

The validation was limited to K and L-shell ionization, for which extensive compilations of experimental measurements are available in the literature; the scarcity of experimental data for M-shell ionization prevented a similar analysis. The validation process adopted the same method as documented in [29]. Due to the complexity of the subject, only the main conclusions relevant to the assessment of validity of ISICSoo are summarized here; the detailed results are documented in a wider study [30] dedicated to the evaluation of open source codes for the calculation of ionization cross sections by proton impact.

For each target element, cross sections were calculated at the same energies as the experimental data for all the various ECPSSR options; for the K-shell, cross sections were also calculated for the combined Hartree-Slater, United-Atom and Relativistic-Projectile-Velocity options. The compatibility of the calculated cross sections with measurements was determined by means of the χ^2 test; a 0.05 significance was set to reject the null hypothesis of compatibility between calculation and experiment.

For K-shell ionization, the null hypothesis is accepted with 0.05 significance in a fraction of test cases going from 67 ± 6% to 77 ± 5%, depending on the ECPSSR modeling option; the highest fraction of compatible test cases corresponds to the ECPSSR with Hartree-Slater correction configuration. For L-shell ionization, the best compatibility is achieved by cross sections calculated by ECPSSR with United-Atom option; for this configuration the null hypothesis is accepted with 0.05 significance in 79 ± 4% of the test cases over the three L_1, L_2 and L_3 sub-shells.

The validation process involved not only ISICSoo, but also two other freely available software systems for the calculation of ionization cross sections based on the ECPSSR theory mentioned in section 1: ERCS08 and Šmit's code. ISICSoo is found to achieve the highest compatibility with experimental data over the whole set of K- and L-shell test cases; the detailed results of the comparative evaluation of the accuracy of the three codes are documented in reference [30].

The validation process also evaluated possible systematic effects in the cross sections due to the values of atomic electron binding energies used in the calculation. This analysis examined the cross section accuracy associated with the two options of binding energies distributed with ISICS, the compilations by

Bearden and Burr [31] and by Williams [32], and with other compilations [33, 34, 35, 36, 37] used by the GUPIX PIXE analysis code [38] and by general purpose Monte Carlo systems for particle transport. Larger discrepancies with respect to experimental data for K-shell are observed when using the atomic binding energies collected in the Evaluated Atomic Data Library (EADL) [33], while the cross section accuracy achieved with the other compilations is statistically equivalent. The full set of results are reported in detail in a wider scope paper [39], which documents a survey of atomic binding energies compilations used by major Monte Carlo transport systems.

The results of the ISICSoo validation process are also relevant to the original ISICS, since the software verification process described in the previous section established the equivalence of the cross section calculated by ISICS and ISICSoo. It is worthwhile to note that the validation of ISICSoo is more extensive than that of ISICS: for example, although recent ISICS versions offer two options of atomic binding energies, their relative effects on cross section accuracy were not previously documented in association with ISICS. Users of both codes – ISICSoo and original ISICS – can profit from the outcome of the ISICSoo validation process to optimize their choice of run configuration options depending on their experimental requirements.

6. Computational environment and performance

The only prerequisite to use the ISICSoo class is a C++ compiler. The code has been tested on three platforms: Scientific Linux 5 with the gcc 4.1.2 compiler, Mac OS X 10.6.5 with gcc 4.2.1 and Windows XP with MS Visual C++ 2010 Express.

Using -O2 optimization with gcc version 4.1.2 took approximately 3 times longer than compiling without optimization (about 6 seconds versus 2 seconds), but resulted in about 25% shorter running time.

Some timing tests have been performed on an AMD 2.4 GHz 2-core processor machine, the results of which are shown in Figure 2. There is no dependence on energy nor target or projectile atomic number. The only changes in performance are due to larger Gauss-Legendre quadrature order (n) used in integration and the ionized shell. The calculation of L-shell is about 3 times longer than the calculation for K-shells, while for M-shell the calculation takes approximately 10 times longer than for K-shell.
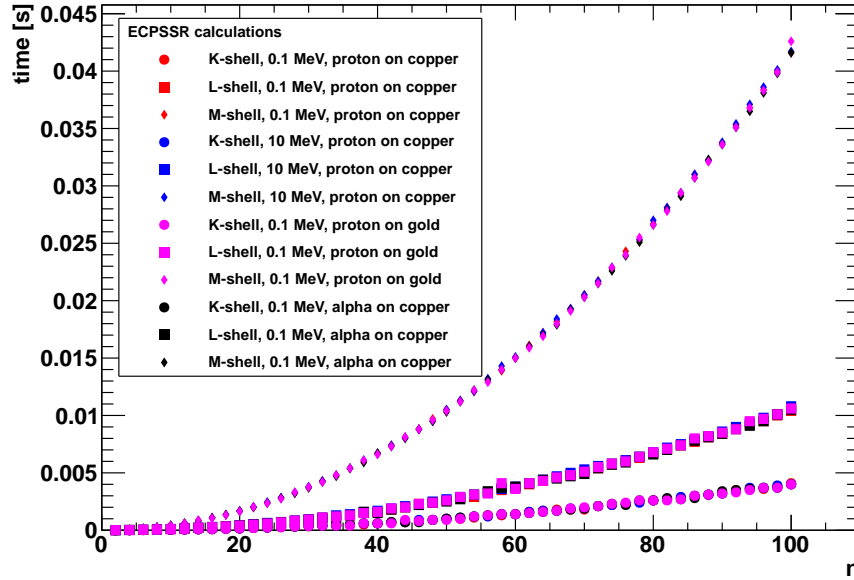

Figure 2: Time to complete the calculation for various projectiles, targets, projectile energies and shells with respect to the Gauss-Legendre quadrature order. Circles are for K-shell, squares for L-shell and diamonds for M-shell calculations. Colors correspond to various projectiles, targets and energies, but the plot clearly shows that the calculations do not depend on them.

Figure 3 shows how the calculated cross-section converge with increasing Gauss-Legendre quadrature order. The default quadrature order ($n = 50$) is not included in the plot, as the results start converging already at about $n = 20$. Hence, if timing performance is of utmost priority, using $n = 20$ or even $n = 30$ will speed the calculation, while the results will still be sufficiently precise. The default value ($n = 50$) is set in the class constructor; it can be modified through the public interface of ISICSoo. As shown in figure 3, the default value is conservative; if optimization is needed by computationally intensive use cases as a trade-off between accuracy and speed of execution, such a process should be done by the interested user, since it may be affected by the computational environment. However, it should be noted that this parameter affects the precision of the cross section calculation, while the accuracy of cross sections is determined by a variety of reasons, most of which depend on the intrinsic physics capabilities of the theory, rather than on the mathematical precision of the calculation.
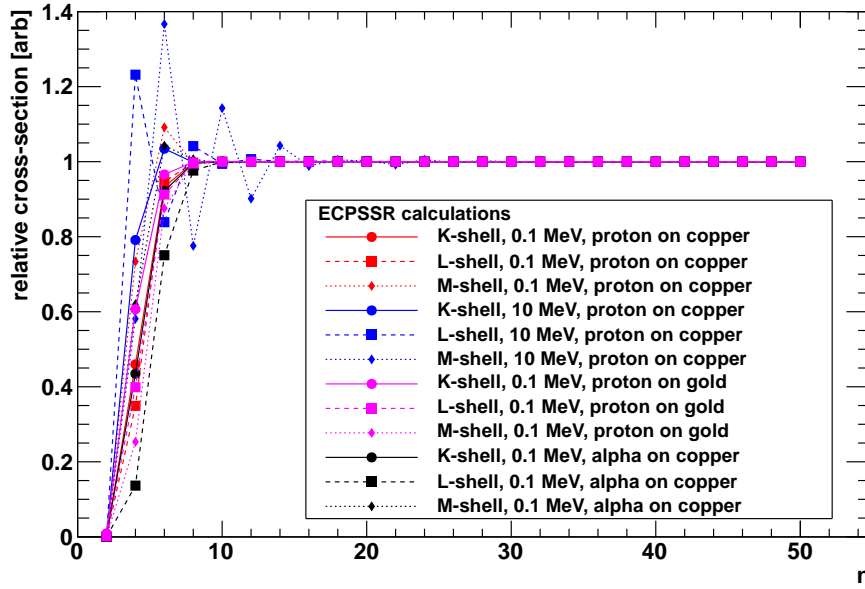

Figure 3: The calculated cross section for ECPSSR calculated with respect to calculations at $n = 100$ show that the results start converging at about $n = 10$, depending on projectile energy.

Using various options of ECPSSR calculations (e.g. Hartree-Slater correction) does not influence the running time significantly.

7. Application example

A simple example of how to use the ISICSoo class is illustrated in Listing 1. It consists of a main function, where an instance `isics` of ISICSoo is created (line 6) and used to calculate the K-shell ECPSSR ionization cross section of 100 keV proton on gold.

Listing 1: Example of a program using ISICS class.

```
1  #include <iostream>
2  #include <ISICSoo.hh>
3  using namespace std;
4
5  int main(){
6    ISICSoo* isics=new ISICSoo();
7
8    isics->LoadData("energy","energy_GW.dat");
9    isics->LoadData("mass","my_mass.dat");
10
```

```
11   isics->SetN(50);
12   isics->SetProjectileZ(1);//proton
13   isics->SetTarget("Au");  //gold
14   isics->SetEnergyUnit(1); //keV
15   isics->SetEnergy(100);   //100 keV
16   isics->SetVerbosity(1);
17
18   isics->CalculateKShell(true);
19   isics->SetHSRScaling(true);
20   isics->RunECPSSR();
21   double result=isics->GetShellECPSSR(0);
22
23   cout<<"K-shell ECPSSR cross-section for "
24       <<isics->GetProjectile()
25       <<" with energy 100 keV on "
26       <<isics->GetTarget()
27       <<" is "
28       <<result<<" barn."<<endl;
29   return 0;
30 }
```

The example shows how to replace default physics data with user selected one in the cross section calculation: here Williams' atomic binding energies (line 8), which are part of ISICS distribution, and user supplied atomic masses (line 9) are loaded. These data files are expected to be in a directory identified through the ISICS_DATA environmental variable, or in the directory where ISICSoo resides, if no environmental variable is specified.

Lines 11-16 set some of the options for running the calculation: the number of Legendre polynomials used in Gaussian quadrature integration (line 11), projectile and target atomic numbers (lines 12 and 13), energy units for energy input (0 for eV, 1 for keV and 2 for MeV) on line 14 and energy of the projectile on line 15. As described previously, the SetVerbosity function on line 16 controls the textual output of the calculations. On line 18 the shell, for which calculations will be done, is chosen; in this example this is the K-shell (CalculateKShell(true)). The user may choose to calculate just one or several shells at the same time.

The following line 19 selects the Hartree-Slater scaling option for ECPSSR calculation.

Alternatively, the settings defined through ISICSoo public interface can be loaded from a configuration file by replacing lines 8-19 with isics->ReadConfig("test.cfg"), where the function argument corresponds to the file to be loaded. The configuration file corresponding to the settings in this example is shown in the code excerpt 2.

The calculation of the ECPSSR model is initiated on line 20. The RunECPSSR() member function will run the ECPSSR calculations and populate the output data holders, one of which is then retrieved on line 21, getting the ECPSSR cross section for K-shell (index 0).

Listing 2: An example of a config file.
```
number of points for gauss legendre quadrature 50,
; this is comment
projectile atomic number 1, ;proton
target atomic number 79,      ;gold
projectile energy unit 1 (keV),
start energy 100 keV,
end energy 101 keV,
energy step 5 keV,
calculate K shell 1 (yes),
calculate L shell 0 (no),
calculate M shell 0 (no),
relativistic projectile velocity 0 (no),
united atom approximation 0 (no),
hsR scaling for K shell 1 (yes),
Run it (no) 0 ; I will call Run myself
```

Further examples are available in the examples folder of the package distributed in the CPC Program Library .

8. Conclusions

ISICS has been reengineered into a C++ class for easier portability across different operating systems and to exploit its physics functionality within other software systems, rather than as a stand-alone program.

Apart from the change in the software design, the new code includes some modifications to the software implementation, which improve the robustness of the results and the computational performance.

A rigorous test process has verified the equivalence of the reengineered version with respect to the previous one (apart from the above mentioned improvements) and has evaluated the compatibility of ISICSoo calculations with an extensive collection of experimental data. The implementation improvements originally developed in ISICSoo have been introduced in the ISICS 2011 version to ensure equivalent functionality of the stand-alone code on Windows platforms with respect to ISICSoo. The achievable accuracy has also been favorably compared to the results of other similar codes for the computation of ECPSSR cross sections.

The reengineered version of ISICS has been successfully exploited to produce the PIXE data library [40] distributed by RSICC, which is used for PIXE simulation [29] in the Geant4 toolkit [24, 25].

Concepts and methods exploited in reengineering ISICS into ISICSoo could be also useful to the evolution of other stand-alone programs available in the CPC Library into components usable in larger scale software systems.

References

[1] Z. Liu and S. J. Cipolla, Comp. Phys. Comm. 97 (1996) 315-330.

[2] S. J. Cipolla, Comp. Phys. Comm. 176 (2007) 157-159.

[3] S. J. Cipolla, Comp. Phys. Comm. 180 (9) (2009) 1716-1717.

[4] S. J. Cipolla, Comp. Phys. Comm. 182 (2011) 2439-2440.

[5] W. Brandt and G. Lapicki, Phys. Rev. A 23 (1981) 1717-1729.

[6] E. Merzbacher and H. Lewis, Handbuch der Physik 34 (1958) 166-192.

[7] Z. Šmit, Nucl. Instrum. Meth. B 36 (1989) 254-258.

[8] V. Horvat, Comp. Phys. Comm.180 (2009) 995-1003.

[9] https://people.creighton.edu/s̃jc90939/files/ISICS10_GEANT4.zip

[10] S. Demeyer, S. Ducasse, and O. Nierstrasz, "Object-Oriented Reengineering Patterns", Square Bracket Associates, 2008.

[11] M. Fowler, "Refactoring: improving the design of existing code", Addison-Wesley Longman Publishing Co., Inc., Boston, (1999).

[12] Herb Sutter and Andrei Alexandrescu, C++ Coding Standards: 101 Rules, Guidelines, and Best Practices, Addison-Wesley, 2004.

[13] Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Prentice Hall, 2008.

[14] Grady Booch, Object-Oriented Analysis and Design with Applications, Addison-Wesley, 1993.

[15] Rebecca Wirfs-Brock et al., Object Design: Roles, Responsibilities and Collaborations, Addison-Wesley, 2002.

[16] http://www.doxygen.org.

[17] S. J. Cipolla, Nucl. Instrum. Meth. B 261 (2007) 142-144.

[18] G. Lapicki, X-Ray Spectrom. 34 (2005) 269-278, 2005.

[19] G. Lapicki, J. Phys. B 41 (2008) 115201.

[20] ISO/IEC 14882:2003 Programming Language C++ (2003).

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, Numerical Recipes 3rd Ed. (2007), Cambridge University Press, New York, NY, USA.

[22] I. Jacobson, G. Booch, and J. Rumbaugh, "The Unified Software Development Process", Addison-Wesley, 1999.

[23] IEEE Standard for Software Verification and Validation", IEEE Std 1012-2004, (2004).

[24] S. Agostinelli et al., Nucl. Instrum. Meth. A 506 (2003) 250-303.

[25] J. Allison et al., IEEE Trans. Nucl. Sci. 53 (2006) 270-278.

[26] H. Paul and J. Sacher, "Fitted empirical reference cross sections for K-shell ionization by protons", *At. Data Nucl. Data Tab.*, vol. 42, pp. 105-156, 1989.

[27] R. S. Sokhi and D. Crumpton, "Experimental L-Shell X-Ray Production and Ionization Cross Sections for Proton Impact", *At. Data Nucl. Data Tables*, vol. 30, pp. 49-124, 1984.

[28] I. Orlic, J. Sow, and S.M. Tang, "Experimental L-shell X-ray production and ionization cross sections for proton impact", *At. Data Nucl. Data Tables* 56 (1994) 159-210.

[29] M. G. Pia et al, "PIXE simulation with Geant4", IEEE Trans. Nucl. Sci., 56 (2009) 3614-3649.

[30] M. Batic, M. G. Pia and P. Saracco, Validation of proton ionization cross section generators for Monte Carlo particle transport, submitted to IEEE Trans. Nucl. Sci. (2011).

[31] J. A. Bearden and A. F. Burr, "Reevaluation of X-Ray Atomic Energy Levels", *Rev. Mod. Phys.*, vol. 39, pp. 125142, 1967.

[32] A. C. Thompson et al., "X-ray Data Booklet", 3rd ed., Lawrence Berkeley Natl. Lab., 2009.

[33] S. T. Perkins et al., "Tables and Graphs of Atomic Subshell and Relaxation Data Derived from the LLNL Evaluated Atomic Data Library (EADL)", Z=1-100, UCRL-50400 Vol. 30, 1997.

[34] T. A. Carlson, "Photoelectron and Auger spectroscopy", Plenum, New York, 1975.

[35] R. B. Firestone and V. S. Shirley eds., "Table of Isotopes, 8th ed.", John Wiley & Sons, New York, 1996.

[36] M. Lederer and V. S. Shirley eds., "Table of Isotopes, 7th ed.", John Wiley & Sons, New York, 1978.

[37] K. D. Sevier, "Atomic electron binding energies", *Atom. Data Nucl. Data Tables*, vol. 24, pp. 323-371, 1979.

[38] J. A. Maxwell, J. L. Campbell, and W. J. Teesdale, "The Guelph PIXE software package", *Nucl. Instrum. Meth. B*, vol. 43, no. 2, pp. 218-230, 1989.

[39] M. G. Pia et al., Evaluation of atomic electron binding energies for Monte Carlo particle transport, submitted to IEEE Trans. Nucl. Sci. (2011).

[40] M. G. Pia et al., PIXE: Proton/alpha Ionisation (K, L, M shell), Tabulated Cross Section Library. Available from the Radiation Safety Information Computational Center (RSICC) at Oak Ridge National Laboratory as DLC-246.